The users of this new language were so enthusiastic about it that IBM developed Fortran compilers for all its stored-program computers, and most of its competitors did the same within only a few years. Applications programmed in Fortran for one machine could readily be converted for use on another, and thus Fortran became the common computer language by which scientists and engineers shared programs and computational processes.

Based on his experience with Fortran, Backus became interested in methods for expressing the syntax, or grammatical rules, of programming languages. He developed a language for this which he introduced at a Unesco conference in 1959. These concepts were improved by Peter Naur (1928–), and the resulting **Backus Naur Form** became the primary methodology for specifying syntax.

BIOGRAPHY

John Backus. Born 3 December 1924 in Philadelphia, Pennsylvania. Received B.S. in mathematics from Columbia University, 1949; A.M. in Mathematics, 1950. Served in U.S. Army, 1942–46. Developed Fortran while employed by IBM, 1954–57. Presented initial concept for Backus Naur Form at Unesco conference on ALGOL 68 in Paris, France, 1959. Named an IBM Fellow, 1963. Recipient of numerous awards and honorary degrees, including National Medal of Science, 1975; ACM Turing Award, 1977; IEEE Computer Society Pioneer Award, 1980; D. Univ., University of York (England), 1985; and Docteur Honoris Causa de l'Université, Université de Nancy, 1989.

SELECTED WRITINGS

Backus, John. "Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs." *Communications of the ACM*, Vol. 21, 1978.

———. "The History of Fortran I, II, and III." *IEEE Annals of the History of Computing*, Vol. 20, No. 4, 1998, pp. 68–78.

Backus, John, and Harlan Herrick. "IBM 701 Speedcoding and Other Automatic-Programming Systems." *ONR Symposium on Automatic Programming for Digital Computers*, ONR, Washington, D.C., 1954.

FURTHER READING

Lee, J. A. N., and Henry Trop, eds. "25th Anniversary of Fortran." Special Issue, *Annals of the History of Computing*, Vol. 6, No. 1, 1984.

Pugh, Emerson W. *Building IBM: Shaping an Industry and Its Technology*. Cambridge, Mass.: MIT Press, 1995.

Shasha, Dennis, and Cathy Lazere. *Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists*. New York: Springer-Verlag, 1997.

—Luanne Johnson

Backus Naur Form

The Backus Naur Form (BNF) is a formal annotation method used to describe the syntax of programming languages. Originally called the *Backus Normal Form*, the BNF was first used for the definition of the programming language **ALGOL** in the late 1950s. **John Backus** (1924–), creator of **Fortran**, served on the ALGOL development committee with Peter Naur (1928–), a European representative. The notation first proposed by Backus was extended and refined by Naur and served as the metalanguage used to describe ALGOL.

Programming languages are built of strings of symbols, in the same way that sentences in a natural language are built of words, and words are built of characters. But in the case of programming languages, the syntax, or possible combinations of words, must be specified precisely so that a computer can later execute an unambiguous program. In the BNF, the possible combinations of words are described by expansion rules. To describe the syntax for a valid U.S. postal address, for example, there would be a name part, a street address, and a zip code part. In BNF this would be expressed by the following rules:

```
<postal-address> ::= <name-part> <street-address>
                     <zip-part>
<name-part>      ::= <name> | <initials>
<street-address> ::= [<apt>] <house-num>
                     <street-name>
<zip-part>       ::= <town-name> "," <state-code>
                     <ZIP-code>
```

The first line explains that the postal address will consist of a concatenation of name part, street address, and ZIP part. The "name-part" is a syntactic category that can be expanded further using the second line. The characters "::=" indicate that this is a definition.

The name part can be a plain name or initials: the symbol "|" means that only one of these alternatives is used. The "street-address" (third rule) can contain an apartment number followed by a house number and a street name. The enclosing square parentheses—"[]"—denote an optional element in the syntax, in this case the apartment number. The "zip-part" (fourth rule) consists of a town name, comma, state code, and ZIP code. Words inside double quotes represent literal words themselves, like the comma above. Additional rules in this example would allow us to refine the syntactic elements further, up to the level of individual letters and digits.

Using this simple method, it is possible to describe any *context-free language*—that is, one in which the syntactic expansion of any element (every line in the BNF) does not depend on extra variables. Many computer languages can be described in this way. There are several variations and extensions of the BNF, including *extended BNF* and *augmented BNF*, which differ only in the types of constructs allowed and therefore the expressiveness of the metalanguage.

—*Raúl Rojas*

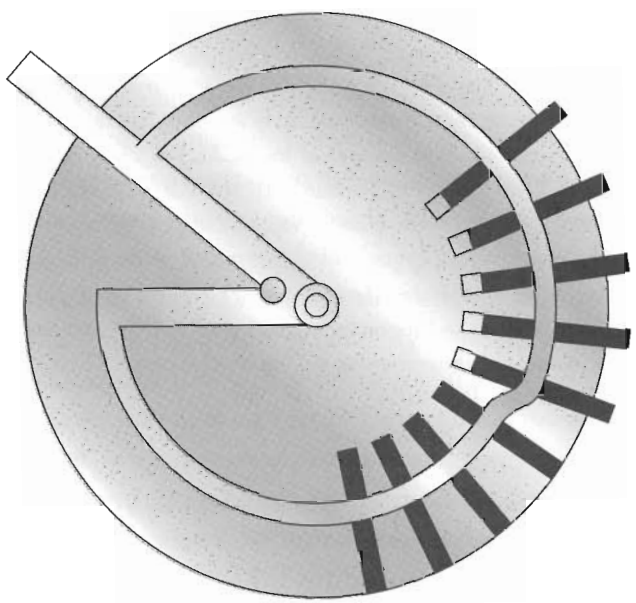
Baldwin–Odhner Calculators

The innovations of the nineteenth century Baldwin–Odhner mechanical calculators the allowed smaller machines to be built. They were all based on the introduction of variable pinwheels. Frank S. Baldwin (1838–1925) and Willgodt T. Odhner developed their calculators from 1872 to 1878. Baldwin's machine used variable-tooth wheels (often referred to as *pinwheels*). He filed a caveat in the U.S. Patent Office in 1872, completed the first machine in 1873, and obtained a patent in 1875. Odhner, who was born in Sweden, invented his own pinwheel calculator in 1874. He produced machines in St. Petersburg, Russia, starting in 1886. The Russian company then sold German production rights to Grimme, Natalis and Co. of Braunschweig, Germany, which went on to produce the famous Brunsviga calculators and sell them on a worldwide basis. Descendants of Willgodt T. Odhner later established the Original Odhner firm in Sweden, which also produced calculators and sold them on a worldwide basis.

The Baldwin–Odhner calculators use a pinwheel for each digit. A simplified drawing of a pinwheel is shown in the figure. As the lever is moved, the number of pins that protrude from the wheel varies from zero to nine. Several of these pinwheels are placed on a common shaft. By setting the various levers, pins representing a multidigit number are raised. Rotating the assembly can advance the wheels of an accumulator by the value of the multidigit number. Multiple rotations advance the accumulator by multiples of the multidigit number. The number of rotations is tracked by a counter register. Two large numbers can be multiplied by shifting the accumulator mechanically.

Rotating the assembly of pinwheels in the opposite direction is used to perform division. In division, the dividend is entered into the accumulator. Then multiples of the divisor are subtracted from the accumulator until the result is approximately zero. The counter register contains the quotient, and the final value in the accumulator is the remainder.

Most desktop electromechanical calculators of the 1940s to the 1960s can be viewed as derived from the Baldwin–Odhner concept. Later versions added a keyboard to enter the operands and a motor.



Simplified illustration of the variable toothed gear.